J Intell Inf Syst (2007) 29:97–110
DOI 10.1007/s10844-006-0033-9

An architecture for electronic field guides

Robert A. Morris · Robert D. Stevenson · William Haber

Published online: 31 January 2007

Abstract People who classify and identify things based on their observable or deducible properties (called "characters" by biologists) can benefit from databases and keys that assist them in naming a specimen. This paper discusses our approach to generating an identification tool based on the field guide concept. Our software accepts character lists either expressed as XML (which biologists rarely provide knowingly—although most databases can now export in XML) or via ODBC connections to the data author's relational database. The software then produces an Electronic Field Guide (EFG) implemented as a collection of Java servlets. The resulting guide answers queries made locally to a backend, or to Internet data sources via http, and returns XML. If, however, the query client requires HTML (e.g., if the EFG is responding to a human-centric browser interface that we or the remote application provides), or if some specialized XML is required, then the EFG forwards the XML to a servlet that applies an XSLT transformation to provide the look and feel that the client application requires. We compare our approach to the architecture of other taxon identification tools. Finally, we discuss how we combine this service with other biodiversity data services on the web to make integrated applications.

Keywords Databases · XML · Electronic Field Guide (EFG)

1 Introduction

Biodiversity software encompasses a wide range of applications including the maintenance of specimen records, analysis of phylogenies, examination of biogeographic relationships, and recording of ecological observations. (See Brach, Lampinen, Liu, and McCree (2000),

R. A. Morris (✉)
Department of Computer Science, University of Massachusetts, Boston, MA, USA
e-mail: ram@cs.umb.edu

R. D. Stevenson
Department of Biology, University of Massachusetts, Boston, MA, USA

W. Haber
Missouri Botanical Garden, St. Louis, MI, USA

Geocities (2004), and Lampinen (2005) for lists of free and commercial packages.) Biodiversity software concentrates at the species level because species are the leaves of the hierarchical grouping system called the "taxonomic tree" used by scientists to classify life forms. Despite limitations to the species concept (Futuyma, 1998; Wheeler & Meier, 2000), the classification system is well established in the scientific community (Winston, 1999) and has been in use since Carl Linnaeus invented it over 250 years ago. Therefore it is not surprising that a significant portion of biodiversity software deals with the management of taxonomic information. Examples include programs that help manage collections of specimens such as Specify (Beach, 2005), Biota (Colwell, 1996), and Biolink (Shattuck & Fitzimmons, 2000), and software with which taxonomists describe and name species, construct keys to differentiate species (See Dallwitz, 1999 for a survey), or document the tree itself (Tree of Life, 2005). Common to all of these programs and many other efforts to share biodiversity information is the descriptive species page (also called the homepage, species summary, or species treatment) in which biologists present basic information about a species.

There are many efforts within scientific and environmental communities to publish data and to establish interoperability and integration standards, practices, and infrastructure. Those efforts include those of the Taxonomic Database Working Group (TDWG, 2005), the US National Science Foundation's Long Term Ecological Research Sites (LTER, 2005), the Citizen Science projects of the Cornell Lab of Ornithology (2003), and efforts of governmental agencies such as the National Biological Information Infrastructure, (NBII, 2003); and the Integrated Taxonomic Information System (ITIS, 2005) and its partners in other countries. Recently, the Global Biodiversity Information Facility (GBIF, 2005), a collaboration of governments, NGOs and research organizations, has been established to coordinate national efforts serving biodiversity information.

Below we describe a biodiversity software application we have developed called the Electronic Field Guide (EFG). It has elements common to biodiversity software listed above but different goals. The UMass Boston EFG Project provides a web-accessible distributed object-oriented database for the identification of biological specimens from field observations. The data, including taxonomic, environmental, and ecological data, aid in identification by building a context for each observation. The EFG project has been expanded to encompass investigation of a number of issues and solutions under discussion in the eco-informatics community, including the use of XML for federation and exchange of data from disparate distributed databases, semantic processing issues, and use of web collaboration tools and of image libraries to expand the resources accessible to an EFG. This paper describes our engineering approach to the building of these systems and reports on their current status. We will discuss specific benefits of object-orientation and of a clean separation between identification and description of species, loosely coupled by a data-driven, service-oriented architecture. Finally, we will show how our EFGs can provide data for integrated biodiversity applications and discuss our future directions.

2 Field guides and descriptive data: object-oriented representations

The correct scientific identification of species is central to the majority of field studies in biology. People learn from others or use field guides if possible, but for most taxonomic groups identification is accomplished with keys constructed by specialists with knowledge and experience in the group. The process begins with the collection of specimens in the field. Taxonomists and systematists then prepare, study, and catalog the specimens, usually in academic institutions or natural history museums. Finally, written descriptions are published and a name given to each new species discovered. Some taxonomists work only in the laboratory, obtaining the specimens from field collectors. Paleontological and, more recently, molecular methods have given rise to the organization of species into taxonomic hierarchies reflecting *phylogenetic hypotheses*, that is, scientific hypotheses about the evolution of species. However, the Linnaean system mostly makes distinctions based on differences visible to the unaided eye or with modern laboratory instruments. Each node in the Linnaean hierarchy is called a *taxon* (plural *taxa*), and the position in the tree is called the *taxonomic rank* of the taxon. For example, the taxon *Vertebrata* at the rank Subphylum, contains all animals with spinal cords. The rank Subphylum has children of taxonomic rank Class, which includes the familiar Classes *Amphibia, Reptilia*, and *Mammalia* (taxa whose common names are close to their Latin names), *Aves* (birds), and three less familiar Classes distinguishing three types of fish: *Agnatha* (fish lacking jaws, such as the Lampreys), *Chondrichthyes* (cartilaginous fish, such as sharks), and *Osteichthyes* (bony fish).

Most paper field guides are devoted either to a specific collection of taxa, e.g., birds, trees, wild-flowers, etc., or to a specific geographic region. In most cases, a field guide user who is interested in ecological interactions will require several different field guides. For example, a guide to butterflies may have some narrative identifying the host and nectar plants of a particular butterfly species, but it would give little help in identifying those plants (which might in turn help the reader identify the butterfly). A field guide on contemporary computers (or the web) can easily hold data on a wide variety of taxa, but a data representation issue immediately arises: descriptive characters appropriate to one group of taxa may have little to do with a radically different group. For example, plants have no wing spots since they have no wings, and butterflies have no leaves to be characterized as simple or compound. Therefore, to represent both organisms in a traditional relational database one must either accept large sparse tables or manage very complex joins on the ecological relationships. Essentially, the diversity of life is not amenable to a single description.

Both Object-Oriented Database Management Systems (OODBMSs, see Loomis & Chaudhri, 1998) and native XML stores (e.g., the open source system eXist (Meier, 2005) are solutions to the problem of representation of biodiversity, in part because their data are hierarchical and, in the case of XML, because the data are self-describing. Gautier, Pave, and Rechenmann (1993) used object orientation to model body part decomposition for identification, and Diederich and Milton (1993) use object orientation to facilitate modularization of taxonomic tools. Saarenmaa, Leppäjärvi, Perttunen, and Saarikko (1995) have observed in detail how object-oriented techniques in general and OODBMS in particular, are well suited to taxonomic representation. However, they reported that, with technologies then available, they were unable to usefully model taxa as classes, rather than as instances of a single class. In a database with a huge number of taxa, class loading overhead would still make this impractical today. Singleton modeling with one class per taxon does not scale to the 1.75 million species that have been described so far (May, 1990; Wilson & Peter, 1988), let alone the estimated 30–100 million believed to be found on earth (Erwin 1988, 1997, May, 1990; Wilson & Peter, 1988), but see Novotny et al. (2002) for lower estimates. Instead, we model an author's treatments of a large group, e.g., a Family, as a Java class, and model individual taxa as instances. The cost of this is that species described in several different treatments, e.g., by different authors for different locations, are not in the same object-oriented class.

We have implemented the UMass Boston Electronic Field Guide on eXcelon Corporation's (now a division of Progress Software) Object Store product. This OODBMS is a persistent store for Java (and C++) classes, and we describe next how we create such classes, along with some of our design principles for author-friendliness, e.g., strategies by which the authors of descriptive data and keys are kept isolated from the technology. We note in passing that object/relational database management systems (Stonebraker et al., 1999) and native XML stores such as eXist now may well provide the support we require, but did not have mature programming interfaces at the time we began coding. Since the original submission of this work, we have also implemented a strategy wherein the object orientation is provided by the web itself, and small flat collections of taxa can be kept in simple relational databases with links to other such collections implemented as web services. We have implemented this using an implementation of the Java Database Connection (JDBC) standard (Sun Microsystems, 2005) accessing a MySQL (2005) database.

We represent a taxon as an instance of a Java class that has little more than a locally unique identifier and a Java HashMap whose keys are the character names and whose values are the character values for the given taxon. This structure is easily serialized both into XML and into an OODB. (The topic of globally unique taxonomic names is an important one, beyond the scope of this paper, although we touch on it briefly below. Suffice it to say that the scientific name of a species is not globally unique over time because new scientific evidence sometimes changes the scope to which names apply or invalidates them altogether. (See Ytow, Morse, and Roberts (2001) or Ytow (2002) for a survey.)

A design requirement of the EFG is that the system should be fundamentally ignorant of the details of descriptive data. If an author chose to offer characters of restaurants of Boston, our software would produce a meaningful and useful restaurant guide instead of a field guide to the butterflies of Costa Rica, our initial target (http://efg.cs.umb.edu/efg). This transparency is important due to the wide variety of choices of characters even among biologists studying the same taxa. However, "official" characters must be supported as a special case of this. More precisely, new species are given names by acceptance in a journal of a description of the species that includes diagnostic characters that distinguish it from previously named species. The author deposits in a collection (usually in a museum or similar institution) a single specimen—called the "type specimen"—that exhibits the published character values. The same or subsequent field expeditions may result in many more specimens of the same species, and by some estimates, there are three billion specimen records in the world's 6,500 natural history museums, not counting microorganisms (Nature, 1998). A small but growing fraction of these records (about 70 million as this article went to press in April 2005) are available through a distributed information system operated by GBIF. Nevertheless, specimen records themselves rarely carry the diagnostic information necessary for the production of keys or field guides. Those are provided by an expert and/or derived from the taxonomic literature. Several projects are underway to use automated or semi-automated techniques to extract characters from digitized legacy paper literature (Heidorn, 2001; Moritz, 2003, personal communication; Vanel, 2005). These will provide legacy "taxonomic treatments" in XML form and should be suitable for the generation of field guides.

Our point of view is shared by the proposed TDWG XML Schema for the Structure of Descriptive Data (SDD, 2005), in whose development the first author participates. Blindness to any particular set of characters or their possible values is a fundamental feature of the design of that Schema. It provides a mechanism whereby an exchange document contains both definitions of the characters and the data permitting those descriptions to support identification of the taxa that the document describes. With early versions of the Schema our system could emit conforming XML and we expect that the EFG will be able to use the final SDD standard for interchange of identification keys and species pages with other software. Agnosticism about characters means that an author cannot entangle her domain expertise with an EFG's internal syntax, which is why our software could produce a restaurant guide as easily as a butterfly guide.

3 Generating EFGs: importing data guided by metadata

We do not require an author to understand XML or object-oriented technologies. Most biologists keep their descriptive data in software that rests on standard table-based databases or in a spreadsheet. Typical systems are Excel, Microsoft Access, FileMaker, and specialized systems such as Lucid (Thiele, 2005) and Delta Access (Hagedorn, 2005). Many of these products can now export XML, and most can respond to SQL queries along an ODBC connection and in turn can be accessed in Java by a JDBC-ODBC bridge or directly via JDBC (Sun Microsystems, 2005). When the database does not emit XML, we use these mechanisms with Java XML support to construct the XML that is the import representation.

Our import strategy is convenient, but leads to several problems. Most notably, the order of delivery of fields is not determined by the ODBC protocol. Typically the order is that in which the fields were created, but that is often not the order in which the biologist may wish them rendered by an application. We address this and other import-related issues in XML metadata, most of which normally is quite stable over the activity of any given biologist and so doesn't impose much technical burden. (To ease this burden further, we investigated the extensibility of the Morpho metadata editor (Higgins, Berkley, & Jones, 2002), which addresses some, but not all, of the issues we face. As a result, we have handcrafted small Java programs to provide the author control over the metadata.) To address character rendering order, the metadata supports an integer weight that may be used as advice to the rendering agent, which can request the metadata along with the data.

A second difficulty is that it is possible to specify field names in many databases that do not yield legal XML identifiers. The metadata contains a mapping of the name in the biologist's data to a name conforming to XML identifier requirements. Other metadata on a per-character basis presently includes information about the data type of the character values and advice from the biologist whether the field should be displayed by an application rendering a taxon descriptive page. The metadata are used to assign data types to the values in the Java HashMap mentioned earlier. They also allow a representation in the map of key-value pairs that record other aspects of the data author's domain expertise besides that which provides the diagnostic characters. For example, our current butterfly EFG offers links to the EFG describing the larval host plants-those on which the butterflies lay their eggs-for each species, which can be an aid in identifying the butterfly. This is because some species very similar in appearance have different preferred larval host plants. Finally, in support of limited structure (e.g., simple lists) in low-end databases and spreadsheets, the data provider may define several list separators, and regular expressions lending syntax to these are also part of the metadata.

Taken together, the import, the key, and the metadata mechanisms have enabled us to build a field guide *generator*, as opposed to a simple field guide. More precisely, our software is agnostic about the semantics of the attributes encoded in the XML (or DBMS) offered by the data author. It has been used to produce EFGs to the Ithomid butterfly Family of Costa Rica and the plants on which they feed or lay eggs, a guide to the aquatic invertebrates of Massachusetts, a key to the plant families and one to the morning glory species of the Monteverde, Costa Rica Cloud Forest, and a winter key to the invasive plants of New England. The identification keys are completely decoupled from the descriptive data in an EFG, which can stand alone or be accessed as a web service from keys, as can other resources such as image libraries. The detailed key system architecture is described elsewhere in this issue. [Editors: need to refer to our "Database backed decision trees" article here]

In early versions, we also supported an embedded taxonomic tree browser. Using metadata supplied by the biologist we attached the imported data into this tree. Because our current architecture supports integrated distributed applications, we instead now build integrated applications that use EFGs as a provider and take taxonomic data from taxonomic name authority servers such as ITIS, 2005). We describe these mechanisms in a later section.

4 The architecture of generated EFGs

Once generated as described above, an EFG is a multi-tier Web application with Java servlets and Java Server Pages (JSP) as middleware forwarding queries to one or more data providers, including a local ObjectStore OODB. ObjectStore, a commercial OODB from the eXcelon division of Progress Software, acts as a persistent Java store for the taxa represented as Java objects in a running EFG. HTML forms, JSP, and Java Applets construct queries which are marshaled by a servlet (Fig. 1). Typically the servlet constructs a query to the backend or other sources[1] and awaits delivery of XML either directly from the data source or, in the case of ObjectStore, by a class that mediates between the Java interface to ObjectStore and the Java Document Object Model (JDOM, 2005) API to XML. Many databases can now return XML directly, no matter what the native storage format. If the remote client did not request transformation (e.g., to HTML or XML to a different schema than the default one, or to some other format such as PDF), we simply return the descriptive data as it was serialized. Otherwise, it is sent to a servlet executing the Xalan XSLT processor (Apache, 2005b) against an XSLT style sheet. (XSLT, the eXtensible Stylesheet Language Transformation specification (W3C, 1999b), is an XML-based language for transforming XML into other XML or into other formats altogether). By this mechanism we support multiple renderings of the responses for different communities, but this can also be accomplished remotely by the client receiving the XML. At this writing we are developing style sheets that support multilingual presentations.

The process of forwarding to an XSLT processor is slightly oversimplified above. In fact, the XML returned to the query engine is forwarded to a small Java Server Page (JSP) and it is that page which forwards to the XSLT processor. An EFG can provide a more or less arbitrary web service, simply by arranging that the JSP page takes some other action on the XML it receives, e.g., wrapping it in SOAP (W3C, 2003). We discuss this below in the section on Integrated Thin Client Applications.

[1] This is a slight oversimplification. See the section on integrated applications.

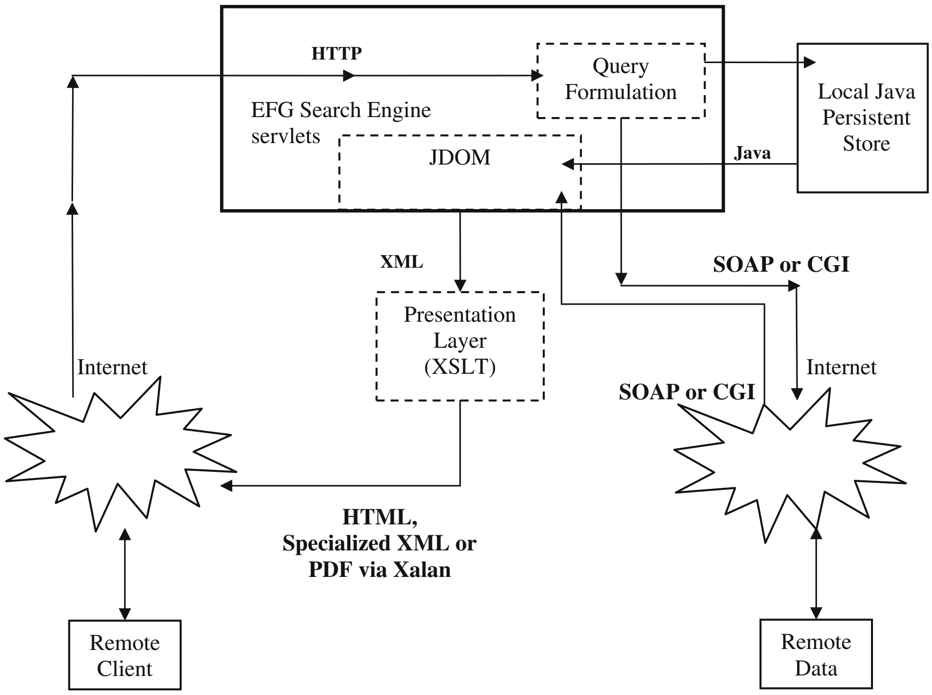
HTTP
Query Formulation
EFG Search Engine servlets
JDOM
Java
Local Java Persistent Store
XML
SOAP or CGI
Presentation Layer (XSLT)
Internet
SOAP or CGI
Internet
HTML, Specialized XML or PDF via Xalan
Remote Client
Remote Data

Fig. 1 EFG architecture. The engine communicates with the data store or remote services after formulating a query in a suitable representation. The data source provides data which the Engine represents in the JDOM API, then serializes it to the presentation layer for transformation into various possible formats. In work underway the local Java store can be proxied by a wrapper around non-object-oriented databases. When the data sources are flat, the object orientation can be implemented by recursively implementing a tree of the services with the architecture of the Figure

Although we operate our ObjectStore and servlets on the same host, this is completely unnecessary. In addition, ObjectStore is itself a distributed client-server system and various pieces of the database could be scattered around the Internet with no change to our architecture except to add resource discovery mechanisms to find the distributed data. In a symmetric fashion, because servlets accept connections on internet IP ports, other clients than our own front end can forward queries to our servlets.

5 User interfaces; taxonomic identification keys decorated by property lists

Our multi-tier interface allows us to support a variety of user interaction. We presently have two interfaces in place, requiring different levels of biological sophistication.

First, we have a pure HTML form-based character-value interface (attributes and values in database terminology). The forms are constructed on-the-fly from the database, and the middleware builds and serves the form to the browser. Each form field offers the user the possible character states. The user can select the maximum number of species she wishes to have offered that meet the current character value choices, and if the system finds more, the user is invited to limit the output by further choice from another character. In our present implementation we do not illustrate those choices as do a number of systems, but this is not precluded by our architecture. Without such illustrations, this interface is suitable mainly for users with some biological training. This style of interface can generally also be written directly in the low-end database systems used by most biologists to store their descriptive data.

We generate a variety of more sophisticated taxonomic identification keys visualized as trees. These are described in detail in [*Editors: here we cite Database backed decision trees in this issue*]. Their principal flexibility comes from the fact that we attach to each tree node collections of property lists that guide the dynamic generation of the key realization. Typically, when used online, these keys request from EFG web services the generation of descriptive pages for further information about the identified taxon. The biggest issue with identification systems is to verify that the identification is in fact correct. As an aid to this, our generated description pages also contain links to similar taxa so that the user may compare the chosen taxon to ones that the database author warns may be confused with it. The aforementioned accompanying paper illustrates such a page.

6 Backends

Figure 1 shows the component architecture of an Electronic Field Guide. The query formulation component of the SearchEngine servlet is not inherently dedicated to querying ObjectStore, but that is presently the only object-oriented backend we have used and in fact the queries are constructed for ObjectStore in the Java servlet implementing the SearchEngine. In work now underway, the Engine is being refactored to provide a more symmetric architecture. More precisely, the SearchEngine will forward the query in an abstract form (probably as XML) to a JSP page much as it now forwards responses. That JSP page will be responsible for transforming the query into whatever the backend requires in order to return XML valid for an XML Schema for which the rendering components can deliver the response. Thus, changing backends can be done simply by changing this JSP page to something else meeting its interface specification and query rewriting can be managed by the JSP. For simple datasets, we have implemented this with MySQL as a backend and will delegate the object orientation to web services by recursive invocation of the architecture illustrated in Fig. 1.

7 Integrated thin client applications

The amount of biodiversity data on the web has expanded rapidly over the past few years, just as has that of many other sciences. Moreover, the diversity of professional, scientific, and educational concerns among the providers and consumers of these data is as great as the diversity in the data themselves. For example, it is typical of identification software such as our EFGs that the full taxonomic detail about the scientific name of a species is rarely accessible from the application. In those instances where the user needs detail such as the exact placement of the name in the standard taxonomic tree, or the history and authority for the name,[2] an electronic agent should be prepared to discover and present that

[2] As new knowledge about a species, or closely related ones, enters the accepted scientific literature, a species name may become obsolete or may be applied in ways different from in an earlier publication (Ytow, 2002; Ytow et al., 2001).

information without human intervention. Using Web Services and other XML-based frameworks, such as those of Apache, Microsoft.NET, and others, it is possible in a few hours to build and deploy purpose-built thin applications returning integrated and filtered XML from multiple sources. Figure 2 illustrates the architecture of one such application we prototyped. The application requests from an EFG the descriptive data for a particular taxon, and queries ITIS for taxonomic data for the same taxon. It filters the data to provide only the so-called synonyms, which are obsolete—though all too often still in use—names for the taxon, along with the taxonomic authority for the currently accepted scientific name. This is merged into a single XML structure and returned to the original requesting client, which is responsible for rendering it (for example, if the client is, or is controlling, a web browser).

Our purpose here is to discuss the integration of unrelated data sources, whose relationship is in fact provided by the integration. It could instead be provided by an ontology, which can provide more general and powerful integration, (see Santos-Mello, 2000 and Rodríguez-Gianolli & Mylopoulos, 2001 for surveys), and we have begun some work in that direction.

The Castor framework (Exolab, 2005) supports XML databinding to Java. This means that Castor accepts an XML Schema describing the constraints on XML data and produces Java source code that models data valid for the Schema. It also provides classes for unmarshalling (converting XML to Java objects) and marshalling (converting Java objects to XML). The generated code also provides accessors and mutators for the Java data modeling the XML elements, which makes it particularly easy to build Java clients accessing data served as XML. These databinding components represent the interior support of our integrated applications. The Schemas involved in the example are that for the XML returned by an EFG, that returned by the Canadian ITIS servers (also now operated at GBIF), and an integration Schema that consists of EFG descriptive data augmented by that portion of the ITIS data mentioned above. This kind of XML response schema integration is not as deep as is needed more generally for the Semantic Web, nor in this example do we have to mediate queries, because, as is typical, both data sources simply have a query consisting of the taxon name and only the response needs mediation. Although hand-built, this Schema could easily be generated by applying a path-expression language like XPath (W3C, 1999a) to the two data schemas, providing for semi-automatic generation of the databinding modules, but at deployment time, not query time.

The network-facing components are implemented as SOAP services using the Apache Axis framework (Apache, 2005a). The ITIS and EFG services are thin clients of the respective underlying services, each of which has only cgi-style interfaces using the http GET protocol. Each thus provide a SOAP proxy to one of those interfaces. In both cases, this proxy can retrieve both XML and HTML from the underlying services, but Fig. 2 describes a purely XML-oriented service. Retrieving HTML by such arrangements is rarely satisfactory because many HTML documents have local hypertext links in them, which are then interpreted by browsers to refer to data on the proxy server, unless the trouble is taken to rewrite the references to resolve to the original providers. Since those providers may themselves be proxies, the success of such architecture would depend on standards for resolution of the ultimate provider in such a chain, but proxies typically intentionally shield these resources from view.

In its simplest use, the Axis framework requires mainly the construction of a small XML file called the Deployment Descriptor file, which describes the parameters for the service and some facts about the Java class that implements it. Axis then provides for this class to

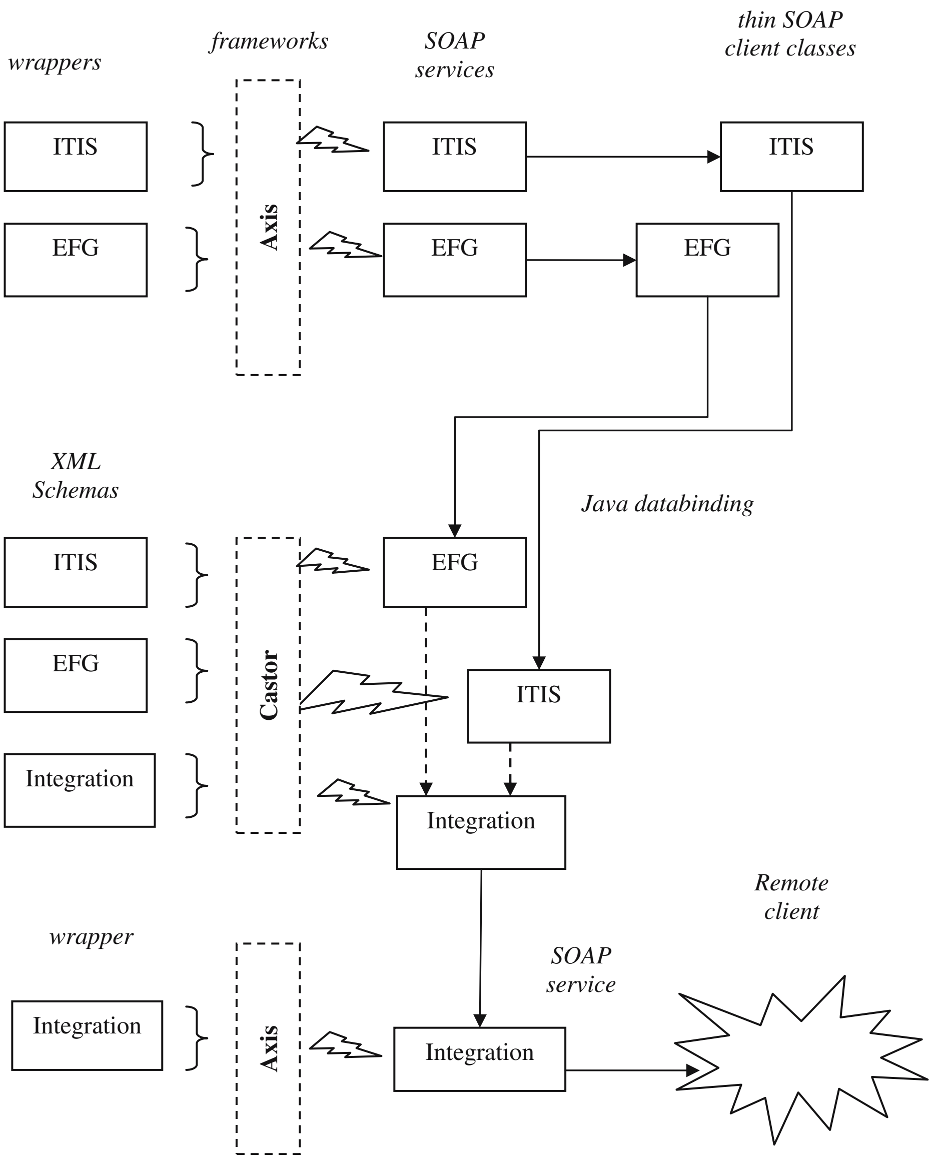

Fig. 2 Data flow in a rapidly produced thin integrated SOAP service. ITIS and EFG wrappers are small Java classes that encapsulate the queries to their respective services. The wrapper schemas are static input to the frameworks, which generate source code for the services and databinding components and also support them in the server architecture (Apache tomcat4). The Integration wrapper has no function except to wrap the Integration databinding output in SOAP. It is actually the network facing service, but we have shown only the dataflow representing returned data, not that representing the query. That is generally the reverse of what we have shown. *Solid arrows* represent data flow as XML, *dotted arrows* as Java objects. *Lightening bolts* denote code generated and compiled at deployment time. See text for details

accept and answer SOAP messages, but it does more. When the service runs under Axis, the underlying web server can also deliver a Web Services Definition Language (WSDL (W3C, 2001)) document to a SOAP client. SOAP clients operating under a suitable framework, including Axis and the Microsoft, 2005) .NET framework, can call the framework to provide the SOAP messaging infrastructure, leaving the SOAP clients to implement mainly the business logic required to deal with the data. In Fig. 2, the ITIS and EFG databinding components are functionally acting as SOAP clients. However, an important part of our architecture not discussed here is the provision of reusable components for building biodiversity applications, combined either by purpose-built applications, as in this example, or by dynamically constructed applications made in a variety of ways, including guided by semantic processing, an area we are just beginning to explore. Therefore, we provide separate Java classes that perform the SOAP interaction, emitting SOAP toward the network, and calling the databinding classes with XML strings. These in turn call the integration databinding component classes to perform the business logic of the integration. The integrated Java object is returned as an XML string to the class implementing the integration SOAP service for communication to the remote application. In our example, the integration SOAP service accepts a taxon name and returns the integrated response of EFG descriptive pages plus ITIS taxonomy. Note that Fig. 2 only shows the flow of responses, not queries.

Finally we should note that WSDL support is especially useful because the Universal, Description, Discovery and Integration (UDDI, 2005) protocols and servers provide powerful resource discovery capabilities to WSDL-based services UDDI (2005), and because clients written against WSDL require a particularly small amount of SOAP-related code when they run under frameworks supporting the Web Services protocols.

8 Relation to other work

Computer applications to systematic biology dates to the early 1970s. (See Pankhurst (1993) for a survey, and Marcus (1993) for an overview of systematic biology), but theoretical aspects (Osborne, 1963) of the graphs underlying keys, as well as practical treatises (Metcalf, 1954) appeared several decades earlier. At about the time that many specialized disciplines were struggling with electronic typesetting (ACM, 1988), Payne (1984) reported on such an effort for identification keys, solving problems that are now routine. By the early 1990s a wide range of software and database technologies had been applied to many problems of systematics (Fortuner, 1993).

Systems like Lucid (Thiele, 2005), uBio (2004), INTKEY (Dallwitz, 1993; Dallwitz, Paine, & Zurcher, 1993), Delta Access (Hagedorn, 2005) and the ID Nature Guides of DiscoverLife (discoverlife.org, 2004) produce “multi-entry keys” in which the system offers its notion (rather than the author’s notion) of the best character to score next. Dallwitz (1974) presented a formal analysis of a cost function for this process. Some systems using this approach can add an author component by supporting a weighting factor to discourage the system from offering difficult to score characters. This kind of system can be very useful for data with thousands of characters, which might be unwieldy to produce in EFG keys. (However, EFG Keys can have subkeys, making it easier to break up the work). EFG keys differ from most other existing key software in leaving the operators in complete control of the server on which web-accessible versions of the keys are offered (uBio’s key generator and scoring browser are open source PHP programs, so can be run on any platform supporting PHP) and in providing, with no additional effort, versions of the key which need no software at all except a browser. The largest difference between these systems and EFG keys is that EFG keys can, but need not be, completely independent of the database in which descriptive data is housed. Each of the aforementioned systems is tied to its own descriptive databases, and depends on importing data, rather than accessing it live, for information from other running descriptive systems. As does our system, uBio keys and the resulting identification provide for arbitrary XSLT style sheets, so can allow a developer to organize differing views of the both key and the result, including a descriptive page based on the taxon-by-character matrix driving the key. We are unaware of any other biodiversity information system than the EFG whose architecture is explicitly designed to support distributed searching for species descriptive data (Fig. 1).

The present EFG software supports a limited multi-entry key in which simple, automatically generated, forms-based access is given to the taxon-by-character matrix underlying the database (if there is one).

Finally, we should mention that methods of Geographic Information Systems (GIS) are now widely pervasive in biodiversity informatics, with geographic distribution based on professional extrapolation of known occurrences, or predictive modeling (Aspinall, Burton, & Landenburger, 1998) and some online field guides, e.g. those of the National Wildlife Federation (2005) provide specialized descriptions of species in particular places.

9 Future work

Ontology-driven XML integration has been widely explored. (See Santos-Mello (2000) and Rodríguez-Gianolli and Mylopoulos (2001) for surveys.). We expect such methods to be valuable in the domain of invasive species, where we are developing an ontology to describe complex relations between invasive species, their impact and control. Internet data on the impact and control of invasive species are particularly heterogeneous and lack any controlled vocabulary. In a second project just beginning, we are exploring semantic methods to examine metadata for remote services with the aim of reducing the impact of network latencies by not querying data sources for which it can be deduced that no response will be of value. Finally, we expect to investigate semantic processing as a mechanism to guide dynamically produced applications from the pieces of our component architecture.

Other work under design includes XML access control of sensitive data (for example about the precise location of an endangered species) and implementation of digital gazetteer components for applications that will allow personalized EFGs to be carried into the field on GPS-equipped handheld computers, keying local names to the GPS data. We are designing more dynamic discovery mechanisms for the distributed query architecture, which presently finds its resources from static lists provided at system configuration. Finally, we have recently begun work using JPEG2000 (2004) image files as data mules to carrying descriptive data about the species depicted in them.

Acknowledgements Robert Morris, Robert Stevenson, and William Haber were partially supported under NSF Grants DBI-9808462 DBI-0113058, and DBI-0111540 from the US National Science Foundation.

Quite a number of present and former students have contributed to the programming effort of the work described here. Among them are Matthew Passell, Elvan Kerim, Jun Wan, Hui Dong, Hua Tang, Jacob Asiedu, and Kimberly Lin. Asiedu has since joined the project as a full-time software engineer.

Additional funding on subcontracts on KDI-9873021 to the Biodiversity Research Center at the University of Kansas (James Beach, PI) and NIH DC04731 to Boston University (David Mountain, PI).

Portions of this work describing the early versions of EFG appeared in a European Environmental Agency Technical Report (Morris et al., 2001).

References

ACM (1988). *Proceedings of the ACM conference on document processing systems*. New York: Association for Computing Machinery.

Apache (2005a). Apache axis. http://ws.apache.org/axis/ [accessed 2005].

Apache (2005b). Xalan-Java version 2.5.0. http://xml.apache.org/xalan-j/ [accessed 2005].

Aspinall, R. J., Burton, G., & Landenburger L. (1998). Mapping and modeling wildlife species distribution for biodiversity management. http://gis.esri.com/library/userconf/proc98/PROCEED/TO800/PAP783/P783.HTM [accessed 2005].

Beach, J. (2005). Specify biodiversity collections management. http://www.specifysoftware.org [accessed 2005].

Brach, A. R., Lampinen, R., Liu, S., & McCree, K. (2000). Internet directory for botany. http://www.botany.net/IDB/ [accessed 2005].

Colwell, R. K. (1996). *Biota: The biodiversity database manager.* Sunderland, Massachusetts: Sinauer Associates.

Cornell Lab of Ornithology (2003). Citizen science. http://www.birds.cornell.edu/LabPrograms/CitSci/ [accessed 2005].

Dallwitz, M. J. (1974). A flexible computer program for generating identification keys. *Systematic Zoology, 23*, 50–57.

Dallwitz (1993). DELTA and INTKEY. In R. Fortuner (Ed.), *Advances in computer methods for systematic biology* (pp. 287–296). Baltimore: The Johns Hopkins University Press.

Dallwitz, M. J. (1999). Desirable attributes for interactive identification programs.

Dallwitz, M. J., Paine, T. A., & Zurcher, E. J. (1993). *User's guide to the DELTA System: A general system for processing taxonomic descriptions* (4th edn.). http://biodiversity.uno.edu/delta/ [accessed 2005].

Diederich & Milton (1993). NEMISYS: A computer perspective. In R. Fortuner (Ed.), *Advances in computer methods for systematic biology* (pp. 165–179). Baltimore: The Johns Hopkins University Press.

discoverlife.org (2004). Discover life web site. http://www.discoverlife.org/ [accessed 2005].

Erwin, T. L. (1988). The tropical forest canopy: the heart of biotic diversity. In E. O. Wilson (Ed.), *Biodiversity* (pp. 123–129). National Academy Press, Washington, D. C.

Erwin, T. L. (1997). Biodiversity at its utmost: tropical forest beetles. In M. L. Reaka-Kudla, D. E. Wilson & E. O. Wilson (Eds.), *Biodiversity II* (pp. 27–40). Joseph Henry Press, Washington, D. C.

Exolab (2005). The Castor project. http://castor.exolab.org/ [accessed 2005].

Fortuner, R. (1993). *Advances in computer methods for systematic biology artificial intelligence, databases, computer vision*. Baltimore: The Johns Hopkins University Press.

Futuyma, D. J. (1998). *Evolutionary biology.* MA: Sinauer Associates.

Gautier, Pave, & Rechenmann (1993). Object-centered representation and fish identification in Antartica. In R. Fortuner (Ed.), *Advances in computer methods for systematic biology* (pp. 181–195). Baltimore: The Johns Hopkins University Press.

GBIF (2005). The global biodiversity information facility. http://www.gbif.org [accessed 2005].

Geocities (2004). Digital taxonomy software. http://www.geocities.com/RainForest/Vines/8695/ [accessed 2005].

Hagedorn, G. (2005). Delta access. http://www.diversitycampus.net/Workbench/Descriptions/index.html [accessed 2005].

Heidorn, P. B. (2001). A tool for multipurpose use of online Flora and Fauna. *First Monday* (*Online*), *6*.

Higgins, D., Berkley, C., & Jones, M. B. (2002). Managing heterogeneous ecological data using morpho. In: *14th International Conference on Scientific and Statistical Database Management* (*SSDBM'02*) (pp. 69–77). IEEE Computer Society.

ITIS (2005). Integrated taxonomic information system. http://www.itis.usda.gov/ [accessed 2005].

JDOM (2005). JDOM. http://www.jdom.org/ [accessed 2005].

JPEG2000 (2004). JPEG 2000. http://www.jpeg.org/jpeg2000/ [accessed 2005].

Lampinen, R. (2005). Cartographic links for botanists. http://www.helsinki.fi/~rlampine/cartogr.html [accessed 2000].

Loomis, M. E. S., & Chaudhri, A. B. (1998). *Object databases in practice*. Upper Saddle River, N.J.: Prentice Hall PTR.

LTER (2005). The U.S. long term ecological research network. http://www.lternet.edu/ [accessed 2005].

Marcus (1993). The goals and methods of systematic biology. In R. Fortuner (Ed.), *Advances in computer methods for systematic biology* (pp. 31–53). Baltimore: The Johns Hopkins University Press.

May, R. M. (1990). How many species? *Philosophical Transactions of the Royal Society of London. B, 330*, 293–304.

Meier, W. M. (2005). Open source XML database. http://exist-db.org/ [accessed 2005].

Metcalf, Z. P. (1954). The construction of keys. *Systematic Zoology, 3*, 38–45.

Microsoft (2005). Basics of .NET. http://www.microsoft.com/Net/Basics.aspx [accessed 2005].

Morris, R. A., Passell, M., Wan, J., Stevenson, R. D., & Haber, W. (2001). *Engineering considerations for biodiversity software*. European Environmental Agency Technical Reports.

MySQL (2005). MySQL developer zone. http://dev.mysql.com/ [accessed 2005].

National Wildlife Federation (2005). eNature online field guides. http://enature.com/guides/select_group.asp [accessed 2005].

Nature (1998). News briefing: Museum research comes off list of endangered species. *Nature, 394*, 115.

NBII (2003). Biocomplexity thesaurus. http://thesaurus.nbii.gov/about.html [accessed.

Novotny, V., Basset, Y., Miller, S. E., Weiblen, G. D., Bremer, B., Cizek, L., et al. (2002). Low host specificity of herbivorous insects in a tropical forest. *Nature, 416*, 841–844.

Osborne, D. V. (1963). Some aspects of the theory of dichotomous keys. *New Phytologist, 62*, 144–160.

Pankhurst (1993). Principles and problems in identification. In R. Fortuner (Ed.), *Advances in computer methods for systematic biology* (pp. 125–136). Baltimore: The Johns Hopkins University Press.

Payne, R. W. (1984). Computer construction and typesetting of identification keys. *New Phytologist, 96*, 631–634.

Rodríguez-Gianolli, P., & Mylopoulos, J. (2001). A semantic approach to XML-based data integration. In H. S. Kunii, A. Solvberg, & S. Jajodia (Eds.), Conceptual modeling—ER 2001 In: *20th International Conference on Conceptual Modeling*. 117 ff. Springer Lecture Notes in Computer Science.

Saarenmaa, H., Leppäjärvi, S., Perttunen, J., & Saarikko, J. (1995). Object-oriented taxonomic biodiversity databases on the World Wide Web. In A. Kempf & H. Saarenmaa (Eds.), Internet applications and electronic information resources in forestry and environmental sciences. In: *Workshop at the European Forest Institute* (pp. 121–129). EFI Proceedings.

Santos-Mello (2000). A mediation layer for integration of XML data sources with ontology support.

SDD (2005). Taxonomic database working group, Subgroup on the Structure of Descriptive Data (SDD). http://www.tdwg.org/sddhome.html [accessed 2005].

Shattuck, S., & Fitzimmons, N. J. (2000). *BioLink® the biodiversity information management system* (p. 3066). Collingwood Victoria: CSIRO Publishing.

Stonebraker, M., Brown, P., & Moore, D. (1999). *Object relational dbms: Tracking the next great wave*. (2nd edn.). Morgan Kaufmann.

Sun Microsystems (2005) JDBC technology. http://java.sun.com/products/jdbc/ [accessed 2005].

TDWG (2005). IUBS taxonomic database working group. http://www.tdwg.org/ [accessed 2005].

Thiele, K. (2005). Lucid 3. http://www.lucidcentral.com/ [accessed 2005].

Tree of Life (2005). Tree of life web project home page. http://tolweb.org/tree/phylogeny.html [accessed 2004].

uBio (2004). uBio home page. http://www.ubio.org/ [accessed 2005].

UDDI (2005). Universal description, discovery and integration of web services. http://www.uddi.org [accessed 2005].

Vanel, J. M. (2005). Worldwide botanical knowledge base search and identification engine with 8467 species of the Flora of China. http://jmvanel.free.fr/protea.html [accessed 2005].

W3C (1999a). XML Path language (XPath) version 1.0 W3C recommendation 16 November 1999. http://www.w3.org/TR/xpath [accessed 2005].

W3C (1999b). XSL Transformations (XSLT) version 1.0; W3C recommendation 16 November 1999. http://www.w3.org/TR/xslt [accessed 2005].

W3C (2001). Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001 http://www.w3.org/TR/2001/NOTE-wsdl-20010315. http://www.w3.org/TR/2001/NOTE-wsdl-20010315 [accessed 2005].

W3C (2003). SOAP version 1.2 part 1: Messaging framework. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/ [accessed 2005].

Wheeler, Q., & Meier, R. (2000). *Species concepts and phylogenetic theory: A debate*. New York: Columbia University Press.

Wilson, E. O., & Peter, F. M. (1988). *National forum on BioDiversity: Biodiversity*. Washington, D.C.: National Academy Press.

Winston, J. E. (1999). *Describing species: Practical taxonomic procedure for biologists*. New York: Columbia University Press.

Ytow, N. (2002). Managing species names. http://www.gbif.org/GBIF_org/prize/lecture2002.htm [accessed 2002].

Ytow, N., Morse, D. R., & Roberts, D. M. D. (2001). Nomencurator: A nomenclatural history model to handle multiple taxonomic views. *Biological Journal of the Linnean Society, 73*, 81–98.